Проектирование отказоустойчивых систем с использованием принципов graceful degradation для KasperskyOS

Сорокин Игорь Александрович, Igor.Sorokin@kaspersky.com
АО «Лаборатория Касперского», Россия, Москва, 125212, Ленинградское шоссе, д.39А, стр.3, БЦ «Олимпия Парк»

1. Мотивация создания

Одно из ключевых требований в современных (и прежде всего автономных) информационных системах (ИС) является надежность, то есть возможность выполнять заданные функции в течении определенного времени и в заданных условиях эксплуатации. ИС включает в себя множество компонент, совокупность которых обеспечивает выполнение требуемых бизнес-функций. Отказ одной или нескольких подсистем может привести к полной остановке выполнения всех функций системы.

При проектировании ИС возникает задача обеспечения возможности выполнения базовых функций даже при отказе одной или нескольких подсистем. В инженерной практике такое свойство ИС называют *graceful degradation (GD)*. Работа ИС осуществляется под управлением операционной системы (ОС). Очевидно, что наиболее полно такое свойство ИС как GD может быть реализовано только с учетом особенностей архитектуры ОС. Рассмотрим каким образом можно обеспечить GD для ИС под управлением KasperskyOS.

2. Подход при проектировании архитектуры ИС для получения свойства GD

Разработка архитектуры начинается с определения цели и основных функций ИС. При этом для достижения свойства GD необходимо выполнить следующие шаги:

№	Шаг	Описание
1	Декомпозиция	Определение списка функций ИС, используемых канал ввода-вывода, механизмов хранения и отображения данных
2	Выявление зависимостей	Построение графа зависимостями между функциями
3	Оценка критичности	Объединение функций в группы в соответствии с их влиянием на цели ИС
4	Выбор стратегий восстановления	Определение того, как может быть восстановлена та или иная группа функций
5	Изоляция	Определение состава компонент (включая user-space драйверы), которые обеспечивают выполнение функций ИС
6	Мониторинг	Определение способов контроля работоспособности компонентов ИС
7	Автоматическое восстановление	Определение способов восстановления работоспособности

Рассмотрим данные шаги более подробно.

Декомпозиция функций зависит от цели, которая ставится при разработке ИС. Очевидно, что функции могут иметь зависимости. Например, функция прием и передача данных невозможна без функций обеспечения каналов ввода-вывода. Поэтому декомпозицию можно считать законченной, когда выявлены все атомарные функции. Атомарной можно считать функцию, которые могут быть отключена или включена

независимо. Для ОС атомарными можно считать функции, которые реализуются user-space драйверами и ядром ОС.

Шаг **выявление зависимости** необходим, т.к. функции, предназначенные для обеспечения целей ИС, требует выполнения более простых функций. Выявление таких зависимостей удобно представить в виде графа, где узел графа представляет функцию, а ребро – зависимость.

Оценка критичности и **стратегия восстановления** для функции позволяет упростить проведение анализа. Оценка проводится на базе заранее определенных категорий. Для определения перечня категорий и стратегии восстановления можно использовать следующую таблицу.

Уровень	**Категория функции**	**Стратегия**	**Действие при отказе**
L0	Critical (критическая)	Restart	Аппаратный или программный перезапуск ИС
L1	Essential (важная)	Local restart	Все ресурсы ИС направлены на восстановление функции без аппаратного перезапуска ИС (остановка обработки данных, перезапуск процессов и т.п.)
L2	Optional (опциональная)	Minimal	Функция недоступна, попытка восстановление проводится по команде или есть вычислительные возможности.
L3	Auxiliary (вспомогательная)	Ignore	Функция недоступна, восстановление только по команде

Следует обратить внимание, что все функции, которые предоставляются ядром ОС и драйверами, которые требуют аппаратной перезапуска устройства в случае отказа, являются критическими (Critical). Для реализации свойства GD в ИС предпочтение следует отдавать микроядерным ОС (KasperskeyOS), т.к. количество функций реализуемых в микроядерной ОС значительно меньше, чем в монолитной.

Изоляция заключается в распределении функций по компонентам. Обычно в архитектуре ОС под компонентом понимают процесс. В связи с очевидными ограничениями ОС невозможно для каждой функции выделить отдельный процесс. Архитектору необходимо выбрать функциональную нагрузку процессов руководствуясь данными, полученными на предыдущих шагах и прежде всего стратегии восстановления. Практически восстановить работоспособность процесса мы можем только его перезапуском. Это означает, что (по возможности) уровень стратегии функций, предоставляемых одним процессом, должным быть одинаковыми или близкими.

На этом шаге следует также проработать механизмы взаимодействия между процессами (inter process communication - IPC). IPC взаимодействия должны быть асинхронными или необходимо использовать очереди сообщений с timeout. В противном случае, зависимый компонент не сможет отменить выполняемый запрос, в случае, когда запрашиваемый сервис находится в аварийном состоянии.

Мониторинг определяет то какие механизмы могут быть использованы для определения отказа. Для устройств это могут быть: аппаратный watchdog (для перезапуска

устройства, если произошел отказ ядра ОС); программный watchdog, основанный на heartbeats и т.п. Для KasperskyOS разработана технология KCFM (Kaspersky Control Flow Monitor) принцип работы которого основан на трассировке системный вызовов каждого процесса и сравнении цепочки вызовов с эталонными (полученными на этапе компиляции исходного кода приложения).

На данном шаге также следует определить и состояние ИС с точки зрения GD. Например,

- Normal - все компоненты работают штатно;
- Degraded - отключены некритичные функции, но основное управление работает;
- Safe - только критический контроль (например, аварийная остановка, минимальная связь);
- Stop – проводится полная остановка с сохранением состояния.

Переход между состояниями проводится по команде подсистемы мониторинга.

Шаг **автоматическое восстановление** необходим при проработке архитектуры для определения механизмов восстановления для выделенных компонент. На этом этапе в зависимости от состояния ИС (определенные на предыдущем шаге) могут быть проработаны различные сценарии восстановления. Например, если после перезапуска user-space драйвера опять следует отказ, то драйвер переводится в состояние «недоступно», а аппаратное устройство (например, логический диск) отключается, а ИС продолжает работать без доступа к этому диску.

Следует обратить внимание и на механизмы логирования и то, как информация логирования будет доставлена для дальнейшего анализа разработчикам.

3. Заключение

Свойство GD является важным при разработке отказоустойчивых систем. Для достижения данного свойства необходимо выполнить достаточно большую аналитическую и архитектурную работу. Однако, в результате возможно создать ИС, которая противостоит отказам и атакам.

В KasperskyOS есть все механизмы и готовые компоненты, которые позволяют создавать отказоустойчивые системы.